WISMO228 TCP/IP AT Commands Manual

WISMO228 WIreless Standard MOdem

WA_DEV_W228_UGD_002
Revision 003
January 29, 2010

SIERRA WIRELESS
HEART OF THE WIRELESS MACHINE®

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where blasting is in progress, where explosive atmospheres may be present, near medical equipment, near life support equipment, or any equipment which may be susceptible to any form of radio interference. In such areas, the Sierra Wireless modem **MUST BE POWERED OFF**. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Copyright

© 2010 Sierra Wireless. All rights reserved.

Trademarks

AirCard® and "Heart of the Wireless Machine®" are filed or registered trademarks of Sierra Wireless. Watcher® is a trademark of Sierra Wireless, registered in the European Community. Sierra Wireless, the Sierra Wireless logo, the red wave design, and the red-tipped antenna are trademarks of Sierra Wireless.

, , ®, inSIM®, "YOU MAKE IT, WE MAKE IT WIRELESS®", WAVECOM®, WISMO®, Wireless Microprocessor®, Wireless CPU®, Open AT® are filed or registered trademarks of Sierra Wireless S.A. in France and/or in other countries.

Windows® is a registered trademark of Microsoft Corporation.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of the respective owners.

Contact Information

Sales Desk:	Phone:	1-604-232-1488
	Hours:	8:00 AM to 5:00 PM Pacific Time
	E-mail:	sales@sierrawireless.com
Post:	Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4	
Fax:	1-604-231-1109	
Web:	www.sierrawireless.com	

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases:

www.sierrawireless.com

Revision History

Index	Date	Updates
001	October 12, 2009	Creation
002	December 9, 2009	Removed WIP_BOPT_DIAL_RINGCOUNT from +WIPBR.
		+WIPOPT: • Removed option types WIP_COPT_BOUND and WIP_COPT_SND_LOWAT. • Changed the value format of WIP_COPT_TTL.
		Updated the UDP datagram information.
		Added a <wait time> unit for Resetting TCP Sockets.
		Updated the action command syntax for +WIPPING.
		Deleted +WIPS from section 3 General Configuration.
		Deleted the note about +WIPS in the +WIPCFG note.
003	January 29, 2010	Updated the <nwrite> parameter desscription for PING command +WIPPING.

Contents

CONTENTS 6
1. INTRODUCTION 9
AT Commands Presentation Rules 9
AT Commands Syntax 9
Command Line 9
Information Responses and Result Codes 10
Logos 10
2. PRINCIPLES 11
Sockets Identification 11
Possible Protocols 12
Number of Sockets 12
3. GENERAL CONFIGURATION 13
IP Stack Handling +WIPCFG 13
Description 13
Syntax 13
Parameters and Defined Values 14
Parameter Storage 17
Possible Errors 18
Examples 19
Notes 20
Bearers Handling +WIPBR 21
Description 21
Syntax 21
Parameters and Defined Values 22
Parameter Storage 25
Possible Errors 25
Examples 26
Notes 27
4. IP PROTOCOL SERVICES 29
Service Creation +WIPCREATE 29
Description 29
Syntax 30

Parameters and Defined Values 32
Parameter Storage 32
Possible Errors 33
Examples 34
Notes 34
Closing a Service +WIPCLOSE 36
Description 36
Syntax 36
Parameters and Defined Values 36
Parameter Storage 37
Possible Errors 37
Examples 37
Notes 37
Service Option Handling +WIPOPT 39
Description 39
Syntax 39
Parameters and Defined Values 40
Parameter Storage 40
Possible Errors 40
Examples 41
Notes 41
5. DATA EXCHANGE FOR PROTOCOL SERVICES 43
Socket Data Exchange +WIPDATA 43
Description 43
Continuous Mode 43
Continuous Transparent Mode 48
Resetting TCP Sockets 48
Syntax 48
Parameters and Defined Values 49
Parameter Storage 50
Possible Errors 50
Examples 51
Notes 52
6. PING SERVICES 56
PING command +WIPPING 56
Description 56

Syntax 56
Parameters and Defined Values 56
Parameter Storage 57
Possible Errors 57
Examples 58

7. EXAMPLES OF APPLICATION 59

TCP Socket 59
TCP Server Socket 59
TCP Client Socket 61
UDP Socket 63
PING 65
Creating a TCP Server, Spawning the Maximum TCP Sockets (for the Configured Server) 66
Creating a Server and Trying to Create a TCP Client/Server on a Reserved Index (reserved by the Server) will Fail 67
Failed Creation of a TCP Client and Failed Creation of a TCP Server with Index Range Containing a TCP Client 69
Creating 8 UDP Sockets, 8 TCP Clients and 4 TCP Servers 70
Trying to Create 8 UDP Sockets, 8 TCP Client Sockets and 4 TCP Server Sockets 73

8. ERROR CODES 75

9. LIST OF ABBREVIATIONS 77

1. Introduction

This manual presents detailed information about the AT command set associated with the Wavecom IP feature.

AT Commands Presentation Rules

The AT commands in this document are presented as follows:

- A *Description* section provides general information on the AT command (or response) behavior.
- A *Syntax* section describes the command and responses syntaxes and all parameter descriptions.
- A *Parameters and Defined Values* section describes all parameters and values.
- A *Parameter Storage* presents the command used to store the parameter value and/or the command used to restore the parameter default value.
- A *Possible Errors* presents the possible error codes that might be returned when the command syntax is transmitted using the wrong parameters.
- An *Examples* section presents the real use of the described command.
- A *Note* section might also be included indicating some remarks about the command use.
- Figures are provided where necessary.

AT Commands Syntax

The AT command format and the default values for their parameters are presented as follows.

Command Line

Commands always start with the standard prefix "AT+WIP" and end with the <CR> character. Optional parameters are shown in brackets [ ].

Example:

AT+WIPcmd=<Param1>[,<Param2>]

<Param2> is optional. When the AT+WIPcmd is executed without <Param2>, the default value of <Param2> is used.

Information Responses and Result Codes

Responses start and end with <CR><LF>, except for the ATV0 DCE response format and the ATQ1 (result code suppression) commands.

- If the command syntax is incorrect, the "ERROR" string is returned.
- If the command syntax is correct but transmitted with the wrong parameters, the "+CME ERROR: <Err>" or "+CMS ERROR: <SmsErr>" strings are returned with adequate error codes if CMEE was previously set to 1. By default, CMEE is set to 0, and the error message is only "ERROR".
- If the command line has been executed successfully, an "OK" string is returned.

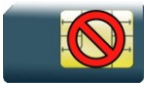
In some cases, such as "AT+CPIN?" or (unsolicited) incoming events, the product does not return the "OK" string as a response.

In the following examples, <CR> and <CR><LF> are intentionally omitted.

Logos

This picture indicates the +WIND indication from which the AT command is allowed. X values can be: 1, 3, 4, 16.

This picture indicates that a SIM card must be inserted to support the AT command.



This picture indicates that an AT command is supported even if the SIM card is absent.

This picture indicates that the PIN 1/CHV 1 code must be entered to support the AT command.

This picture indicates that an AT command is supported even if the PIN 1/CHV 1 code is not entered.

This picture indicates that the PIN 2/CHV 2 code must be entered to support the AT command.

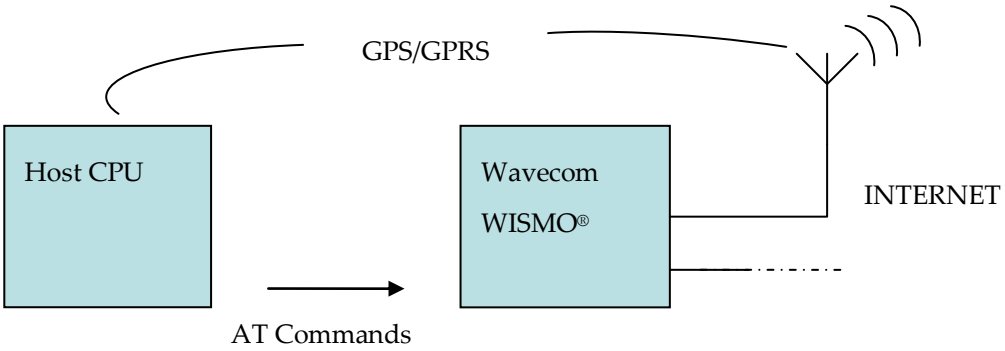
This picture indicates that an AT command is supported even if the PIN 2/CHV 2 code is not entered.

2. Principles

The AT+WIP AT Commands implements the TCP/IP protocols using custom AT commands. The commands are sent from an external application and the corresponding responses are sent back from the WISMO® to the external application.

AT+WIP commands involve:

- a host computer, which issues the AT+WIP commands
- Wavecom's WISMO®
- the rest of the Internet/Intranet



Multiplexing: Several sockets can be operating at once. The +WIPDATA command allows the temporary identification of the UART in data mode with a given socket. The data written on the UART is transferred through the socket. The data which arrives on the socket can be read from the UART.

In AT mode, the host receives an unsolicited event when the data arrives on the socket.

Sockets Identification

Sockets are identified by a pair of numbers: the first one identifies the protocol; the second one identifies a given socket of the protocol.

Possible Protocols

The possible protocols are:

- 1 = UDP
- 2 = TCP in connect mode (Client)
- 3 = TCP in listen mode (Server)

Two pairs with different protocol numbers but the same index identifies two distinct sockets.

Example: Both 1,7 and 2,7 are valid identifiers simultaneously; the former identifies a UDP socket and the later, a TCP connected socket.

Number of Sockets

The number of sockets per protocol is limited and are listed as follows.

- UDP : 8 sockets
- TCP Clients : 8 sockets
- TCP Servers : 4 sockets

3. General Configuration

IP Stack Handling +WIPCFG

Description

The +WIPCFG command is used for performing the following operations:

- start TCP/IP stack
- stop TCP/IP stack
- configuring TCP/IP stack
- displaying version information

Syntax

- if <mode> = 0,1

Action Command **AT+WIPCFG=<mode>** OK

- if <mode>=2

Action Command **AT+WIPCFG=<mode>,<opt num>,<value>** OK

- if <mode>=3

Action Command **AT+WIPCFG=<mode>** TCP/IP AT Command vXX.YY.ZZ OK

- if <mode>=4

Action Command **AT+WIPCFG=<mode>,<action>** OK
Read Command **AT+WIPCFG?** +WIPCFG: <optnum>,<value> [+WIPCFG: <optnum>,<value>[..]] OK
Test Command **AT+WIPCFG=?** OK

Parameters and Defined Values

<mode>:		requested operation
	0	stop TCP/IP stack
	1	start TCP/IP stack
	2	configure TCP/IP stack
	3	display TCP/IP application version
	4	TCP/IP stack configuration management
<opt num>:		configuration option identifier
	0	WIP_NET_OPT_IP_TTL – Default TTL of outgoing data grams This option is a limit on the period of time or number of iterations or transmissions that a unit of data can experience before it should be discarded. The time to live (TTL) is an 8-bit field in the Internet Protocol (IP) header. It is the 9th octet of 20. The default value of this parameter is 64. Its value can be considered as an upper bound on the time that an IP datagram can exist in an internet system. The TTL field is set by the sender of the datagram, and reduced by every host on route to its destination. If the TTL field reaches zero before the datagram arrives at its destination, then the datagram is discarded. This is used to avoid a situation in which an undelivered datagram keeps circulating in the network.

		range: 0-255 (default value: 64)
	1	WIP_NET_OPT_IP_TOS – Default TOS of outgoing parameters The IP protocol provides a facility for the Internet layer to know about the various tradeoffs that should be made for a particular packet. This is required because paths through the Internet vary widely in terms of the quality of service provided. This facility is defined as the "Type of Service" facility, abbreviated as the "TOS facility". The TOS facility is one of the features of the Type of Service octet in the IP datagram header. The Type of Service octet consists of the following three fields: 0 1 2 3 4 5 6 7 +-----+-----+-----+-----+-----+-----+-----+-----+ | | | | | PRECEDENCE | TOS | MBZ | | | | | +-----+-----+-----+-----+-----+-----+-----+-----+ The first field is "PRECEDENCE". It is intended to denote the importance or priority of the datagram. The second field is "TOS" which denotes how the network should maintain the tradeoffs between throughput, delay, reliability, and cost. The last field is "MBZ" (Must Be Zero"), is currently unused and is set to 0. The TOS field can have the following values: 1000 -- minimize delay 0100 -- maximize throughput 0010 -- maximize reliability 0001 -- minimize monetary cost 0000 -- normal service For more information on this field please refer to RFC1349. range: 0-255 (default value: 0)
	2	WIP_NET_OPT_IP_FRAG_TIMEO - Time to live in seconds of incomplete fragments When a datagram's size is larger than the MTU (Maximum Transmission Unit)

	of the network, then the datagram is divided into smaller fragments. These divided fragments are sent separately. This option specifies the Time to live for these fragments. range: 1-65535 (default value: 60)
3	WIP_NET_OPT_TCP_MAXINITWIN – Number of segments of initial TCP window This option is used to specify the number of segments in the initial TCP window. A TCP window specifies the amount of outstanding (unacknowledged by the recipient) data a sender can send on a particular connection before it gets an acknowledgment back from the receiver. The primary reason for the window is congestion control. range: 0-65535 (default value: 0)
4	WIP_NET_OPT_TCP_MIN_MSS - Default MSS of off-link connections This parameter specifies the maximum size of TCP segments which would be sent. By default, the value of this parameter is set to 536. Hence, the firmware would not send any TCP segment having a length greater than 536 bytes without a header. range: 536-1460 (default value: 536)
5	WIP_NET_OPT_DEBUG_PORT This option is used to specify the port on which the debug traces are to be sent. range: 0-3 (default value: 0)
12	AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE - Used for TCP sockets to configure the packet segmentation on the IP network side This option allows the application to buffer a certain amount of data before writing on the IP network side. Each unit in the range represents 100 msec. For example, a value of 10 for this option will give a wait time of 1sec (10 *100mesc). Default value for this option is 0. This value means that no specific process is done to avoid TCP packets segmentation: data are written onto the IP network without any delay after the reception of data. In this case, some TCP packets sent on the IP network may be smaller than TCP_MIN_MSS value. As mentioned above, setting a value of 10 for this option will make the application wait at least 1 second or twice the TCP_MIN_MSS value to be reached before sending data on the IP network. In this case, TCP packets size sent on the IP network should be equal to at least TCP_MIN_MSS (Default value = 536 bytes).

		range: 0-100 (default value: 0)
	13	AT_WIP_NET_OPT_ESC_SEQ_NOT_SENT : Used to configure whether a "+++" escape sequence should be sent as data to the peer. By default, this option is set to 0 which means that the "+++"sequence is sent to the peer as data. If set to 1, "+++"sequence is not sent as data to the peer. range: 0-1(default value:1)
	14	AT_WIP_NET_OPT_AUTO_SWITCH - 0: Does not automatically switch to AT mode 1: Switches automatically to AT mode range: 0-1 (default value:0)
<action>:		requested operation on TCP/IP stack parameter management
	0	stored configuration is erased (restore factory settings)
	1	stores the current configuration parameters
<value>:		value range for different configuration options
<XX.YY.ZZ >:		TCP/IP AT Command release version

Note: UART(WIP_NET_OPT_SOCK_MAX + 1) sockets are reserved when UDP sockets are created (and not for TCP sockets); one socket buffer is added to support/afford DNS accesses.

Parameter Storage

Only one IP stack configuration set can be saved into non-volatile memory.

- "AT+WIPCFG=4,1" is used to store the TCP/IP stack configuration parameters into the non-volatile memory
- "AT+WIPCFG=4,0" is used to free the TCP/IP stack configuration storage

Executing "AT+WIPCFG=1" will apply default parameter values wherever applicable, but it is still possible to change option values at run time using "AT+WIPCFG=2,<optnum>,<optvalue>".

Possible Errors

The possible error message is displayed only if "AT+CMEE=1" is activated. Otherwise, only "ERROR" is displayed.

"+CMEE" AT error code	Description
800	invalid option
801	invalid option value
802	not enough memory
820	error writing configuration in non-volatile memory
821	error freeing configuration in non-volatile memory
844	stack already started
850	unknown reason

Examples

Command	Responses
AT+WIPCFG=1 *Note: Start IP Stack*	OK
AT+WIPCFG?	+WIPCFG: 0,64 +WIPCFG: 1,0 +WIPCFG: 2,60 +WIPCFG: 3,0 +WIPCFG: 4,536 +WIPCFG: 5,0 +WIPCFG: 12,10 +WIPCFG: 13,0 +WIPCFG: 14,0 OK
AT+WIPCFG=2,0,10 *Note: Configure TTL of IP Stack*	OK
AT+WIPCFG?	+WIPCFG: 0,10 +WIPCFG: 1,0 +WIPCFG: 2,60 +WIPCFG: 3,0 +WIPCFG: 4,536 +WIPCFG: 5,0 +WIPCFG: 12,10 +WIPCFG: 13,0 +WIPCFG: 14,0 OK
AT+WIPCFG=3 *Note: Display software version*	TCP/IP AT Commands (release version) OK
AT+WIPCFG=0 *Note: Stop the TCP/IP Stack*	OK
AT+WIPCFG=4,1 *Note: Store IP configuration parameters into non-volatile memory*	OK

Command	Responses
AT+WIPCFG=4,0 *Note: Free IP configuration parameters stored in non-volatile memory*	OK

Notes

It is recommended to change the default settings of the WIP stack using +WIPCFG only when it is required. Changing the parameter values especially the max number of sockets and the max TCP buffer size with high values leads to over consumption of the stack memory which causes the firmware to crash. Hence, care must be taken when the default settings of the stack is changed using the +WIPCFG command.

The following option values set by the +WIPCFG command are taken into consideration at run time. The option values below (except for AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE and AT_WIP_NET_OPT_ESC_SEQ_NOT_SENT) will be taken into consideration at the next start up only if these are saved in non-volatile memory before stopping the stack.

- WIP_NET_OPT_IP_TTL
- WIP_NET_OPT_IP_TOS
- WIP_NET_OPT_IP_FRAG_TIMEO
- WIP_NET_OPT_TCP_MAXINITWIN
- WIP_NET_OPT_TCP_MIN_MSS
- WIP_NET_OPT_DEBUG_PORT
- AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE
- AT_WIP_NET_OPT_ESC_SEQ_NOT_SENT
- AT_WIP_NET_OPT_AUTO_SWITCH

Bearers Handling +WIPBR

Description

The +WIPBR command can be used to:

- select the bearer
- start/close the bearer
- configure different bearer options such as access point name

Syntax

- if <cmdtype> = 0,1 or 5

Action Command

AT+WIPBR=<cmdtype>,<bid>

OK

- if <cmdtype> = 2

Action Command

AT+WIPBR=<cmdtype>,<bid>,<opt num>,<value>

OK

- if <cmdtype> = 3

Action Command

AT+WIPBR=<cmdtype>,<bid>,<opt num>

+WIPBR: <bid>,<opt num>,<value>

OK

- if <cmdtype> = 4

Action Command

AT+WIPBR=<cmdtype>,<bid>,<mode>[,<login>,<password>,[<caller identity>]]

OK

- if <cmdtype> = 6

Action Command **AT+WIPBR=<cmdtype>,<bid>,<mode>** OK
Read Command **AT+WIPBR?** <bid>,<state> [<bid>,<state>[..]] OK
Test Command **AT+WIPBR=?** OK

Parameters and Defined Values

<cmdtype>:		type of command
	0	close bearer
	1	open bearer
	2	set value of different bearer options
	3	get value of different bearer options
	4	start bearer
	5	stop bearer
	6	bearer configuration management
<bid>:		bearer identity
	5	GSM
	6	GPRS
<opt num>:		bearer option identifier
	0	WIP_BOPT_LOGIN – username (string)

		max: 32 characters
	1	WIP_BOPT_PASSWORD – password (string) max: 32 characters
	2	WIP_BOPT_DIAL_PHONENB – phone number (string) max: 32 characters
	7	WIP_BOPT_PPP_PAP - Allow PAP authentication range: 0-1
	8	WIP_BOPT_PPP_CHAP - Allow CHAP authentication range: 0-1
	9	WIP_BOPT_PPP_MSCHAP1 - Allow MSCHAPv1 authentication range: 0-1
	10	WIP_BOPT_PPP_MSCHAP2 - Allow MSCHAPv2 authentication range: 0-1
	11	WIP_BOPT_GPRS_APN - Address of GGSN (string) max: 96 characters
	13	WIP_BOPT_GPRS_HEADERCOMP - Enable PDP header compression range: 0-1
	14	WIP_BOPT_GPRS_DATACOMP - Enable PDP data compression range: 0-1
	15	WIP_BOPT_IP_ADDR - Local IP address (IP/string)
	16	WIP_BOPT_IP_DST_ADDR - Destination IP address (IP/string)
	17	WIP_BOPT_IP_DNS1 - Address of primary DNS server (IP/string)
	18	WIP_BOPT_IP_DNS2 - Address of secondary DNS server (IP/string)
	19	WIP_BOPT_IP_SETDNS - Configure DNS resolver when connection is established range: 0-1
	20	WIP_BOPT_IP_SETGW - Set interface as default gateway when connection is established

		range: 0-1
<value>:		range of values for different bearer options
<mode>:		mode of operation
	0	client
<caller identity>:		optional ASCII string (type ASCII*) If not specified, then target will accept all DATA calls (independent of caller identification). If specified, then target will only accept calls from <caller identity> (which is the GSM data call number of the GSM client).

* IP addresses are displayed in alpha numeric dot format. e.g. 192.168.0.1. When no IP address is known, "0.0.0.0" is displayed.

Caution: ***The options WIP_BOPT_IP_ADDR, WIP_BOPT_IP_DST_ADDR, WIP_BOPT_IP_DNS1 and WIP_BOPT_IP_DNS2 can be read after the bearer connection is established successfully. If an attempt is made to read the options value before the bearer connection is established successfully, an incorrect IP address will be received.***

Parameter Storage

Several bearer configuration sets can be saved.

Calling AT+WIPBR=6,<bid>,1 twice with the same <bid> will store the last configuration set.

- "AT+WIPBR=6,<bid>,1" is used to store the bearer configuration parameters set associated with the bearer <bid> into non-volatile memory.
- "AT+WIPBR=6,<bid>,0" is used to free the bearer configuration parameters set associated with the bearer <bid>.

Executing "AT+WIPBR=1,<bid>" will open bearer <bid> with default parameters of the bearer when existing.

Possible Errors

The possible error message is displayed only if "AT+CMEE=1" is activated. Otherwise, only "ERROR" is displayed.

"+CMEE" AT error code	Description
800	invalid option
801	invalid option value
802	not enough memory left
803	operation not allowed in the current WIP stack state
804	device already open
807	bearer connection failure : line busy
808	bearer connection failure : no answer
814	bearer connection failure : PPP authentication failed
815	bearer connection failure : PPP IPCP negotiation failed
820	error writing configuration in non-volatile memory
821	error freeing configuration in non-volatile memory
847	bearer connection failure: WIP_BOPT_GPRS_TIMEOUT time limit expired before GPRS bearer connected
848	impossible to connect to the bearer
849	connection to the bearer has succeeded but a problem has occurred during the data flow establishment

Examples

Command	Responses
AT+WIPBR?	1,0 6,1 OK *Note: Bearer UART1 is open but not started bearer GPRS is open and started*
AT+WIPBR?	OK *Note: No bearer has been opened yet*
AT+WIPBR=1,6 *Note: Open GPRS bearer*	OK
AT+WIPBR=2,6,11,"APN name" *Note: Set APN of GPRS bearer*	OK
AT+WIPBR=3,6,11 *Note: Get APN of GPRS bearer*	+WIPBR: 6,11,"APN name" OK
AT+WIPBR=4,6,0 *Note: Start GPRS bearer*	OK
AT+WIPBR=5,6 *Note: Stop GPRS bearer*	OK
AT+WIPBR=0,6 *Note: Close GPRS bearer*	OK
AT+WIPBR=1,5 *Note: Open GSM bearer*	OK
AT+WIPBR=2,5,2,"phonenumber" *Note: Set the phonenumber for GSM bearer*	OK
AT+WIPBR=2,5,15,"1.1.1.1" *Note: Set the local IP address for GSM bearer*	OK
AT+WIPBR=2,5,16,"2.2.2.2" *Note: Set the destination IP address for GSM bearer*	OK
AT+WIPBR=3,5,15 *Note: Read the local IP address for GSM bearer*	+WIPBR: 5,15,"0.0.0.0" OK *Note: Local IP address is not set as GSM bearer is still not connected*

Command	Responses
AT+WIPBR=3,5,16 *Note: Read the destination IP address for GSM bearer*	+WIPBR: 5,16,"0.0.0.0" OK *Note: Destination IP address is not set as GSM bearer is still not connected*
AT+WIPBR=4,5,0 *Note: Start the GSM bearer as a client*	OK
AT+WIPBR=3,5,15 *Note: Read the local IP for GSM bearer*	+WIPBR: 5,15,"1.1.1.1" OK
AT+WIPBR=3,5,16 *Note: Read the destination IP for GSM bearer*	+WIPBR: 5,16,"2.2.2.2" OK
AT+WIPBR=5,5 *Note: Stop the GSM bearer*	OK
AT+WIPBR=0,5 *Note: Close the GSM bearer*	OK

Notes

For Starting a Bearer

The mandatory parameters to start a bearer in

- client mode are: <cmdtype>, <bid> and <mode>

Depending on the mode and the bearer type, additional parameters are required or forbidden:

Bid	Mode	Other Parameters
5	0	None
6	0	None

For Opening a Bearer

Opening a bearer only consists of associating the IP protocol stack with the specified bearer. The corresponding bearer setup has to be done through the already existing AT commands.

Several bearers can be opened at the same time but only one bearer can be started at a time.

If both DNS1 and DNS2 are displayed as “0.0.0.0” in the unsolicited message when bearer is opened in server mode, it means that connecting to a remote IP host through a URL will fail.

The options WIP_BOPT_DIAL_REDIALCOUNT and WIP_BOPT_DIAL_REDIALDELAY will not be implemented through AT commands. Nevertheless, for future compatibility reasons, opt num 3 and 4 are kept as reserved.

For GSM bearer, the options WIP_BOPT_IP_ADDR and WIP_BOPT_IP_DST_ADDR will display valid addresses only when the bearer is started and connected, else it will display an address “0.0.0.0”.

4. IP Protocol Services

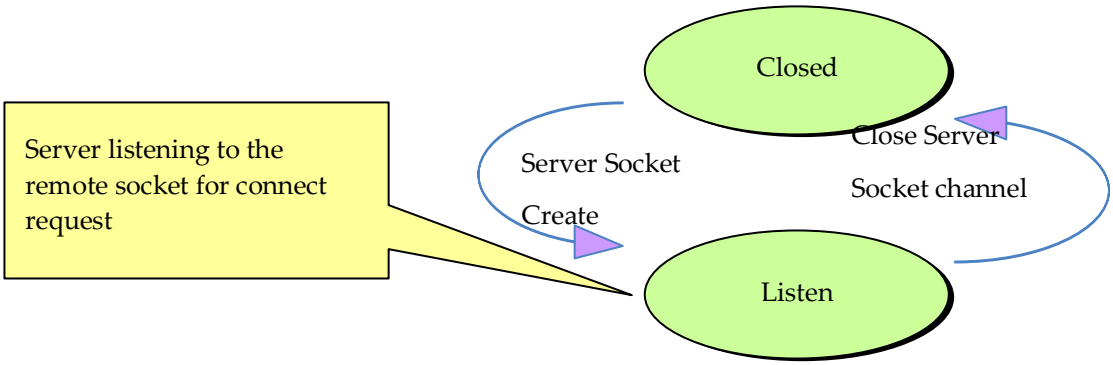
Service Creation +WIPCREATE

Description

The +WIPCREATE command is used to create UDP, TCP client and TCP server sockets associated with the specified index.

If a local port is specified while creating a socket, the created socket will be assigned to this port; if not, a port will be assigned dynamically by the firmware application. If peer IP and peer port is specified, the created socket will be connected to the specified IP and port.

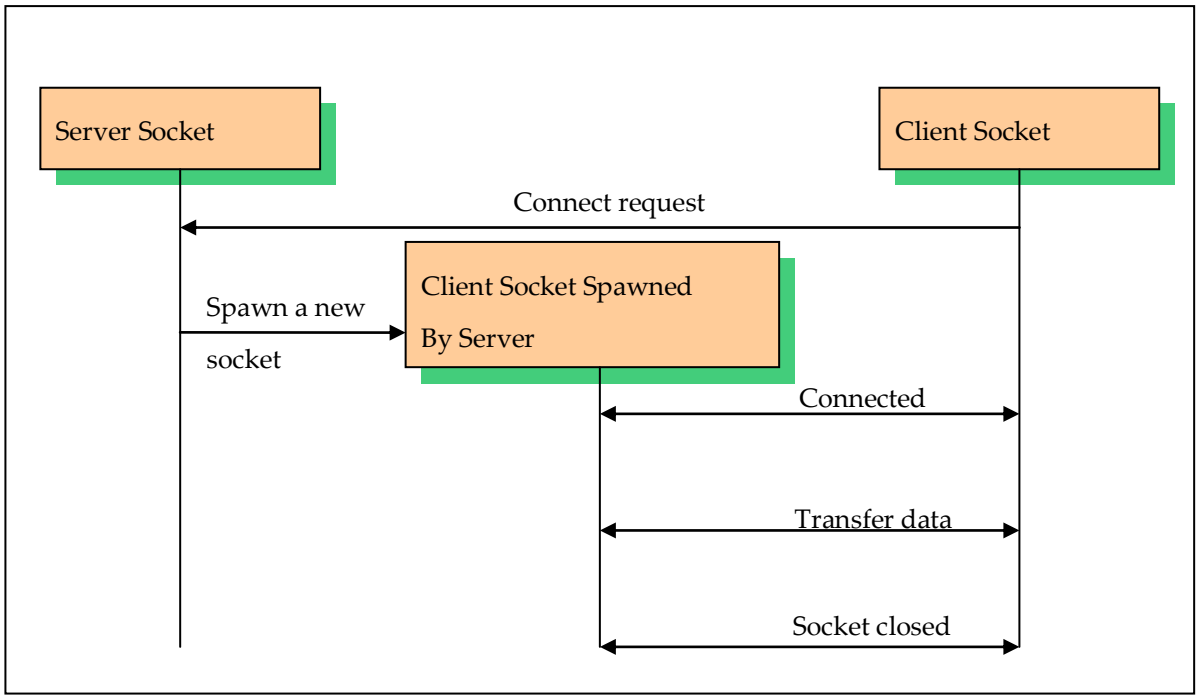
The TCP server cannot be used to transfer data. To transfer data, it creates a local TCP client socket. This process of creating a local socket is referred to as “spawning”. When a server socket is created using spawning, the socket passively listens on a specified port for incoming connections. The diagram below shows the different states managed for a TCP server.



On reception of a connection request from a remote client socket, a server socket does the following:

- spawns a new socket (client) to connect to the remote socket
- transfers data between the spawned socket and the remote socket
- remains in listening mode and is ready to accept requests from other clients

The diagram below shows the connection establishment procedure.



Syntax

- if <mode> = 1

Action Command

AT+WIPCREATE=<mode>,<communication index>,[<local port>] [,<peer IP>,<peer port>]

OK

- if <mode>=2

Action Command

AT+WIPCREATE=<mode>,<communication index>,<peer IP>,<peer port>

OK

- if <mode>=3

Action Command

AT+WIPCREATE=<mode>,<server index>,<local port>,<from idx>,<to idx>

OK

- if <mode>=1 or 2

Unsolicited response

```
+WIPREADY: <mode>,<communication index>
```

- if <mode>=3

Unsolicited response

```
+WIPACCEPT: <server index>,<communication idx>
```

Parameters and Defined Values

<mode>:		Specifies type of socket
	1	UDP
	2	TCP client
	3	TCP server
<index>:		TCP/UDP session identifier
<local port>:		Local TCP/UDP port
<peer IP>:		peer IP address; a string between quotes indicating an address either in numeric form (e.g. “85.12.133.10”) or as a DNS entry (e.g. “www.wavecom.com”)
<peer port>:		peer port or the server port For TCP/UDP, this parameter is the port of the peer socket. range: 1-65535
<from idx>:		minimum index for spawned TCP sockets range: 1-8
<server index>:		TCP server socket identifier range: 1-4
<to idx>:		maximum index for spawned TCP sockets range: 1-8
<communication index>:		indexes reserved for spawned sockets These cannot be used by other sockets even if the spawned sockets are not created yet. range: 1-8

Parameter Storage

None

Possible Errors

The possible error message is displayed only if "AT+CMEE=1" is activated. Otherwise, only "ERROR" is displayed.

"+CMEE" AT error code	Description
800	invalid option
803	operation not allowed in the current TCP/IP stack state
830	bad index
832	bad port number
834	not implemented
836	memory allocation error
837	bad protocol
839	error during channel creation
840	UDP/TCP socket session is already active
842	destination host unreachable (whether host unreachable, Network unreachable, response timeout)
845	attempt is made to reserve/create a client socket which is already reserved/opened by TCP server/client
851	incorrect number of parameters submitted
860	protocol undefined or internal error
865	authentication error
866	server not ready error

Examples

Command	Responses
AT+WIPCREATE=1,1,80 *Note: Create the UDP socket on local port 80 with communication index = 1 ⇔ WISMO® acts as a UDP server awaiting for incoming datagram on local port 80*	OK *Note: An unsolicited event +WIPREADY: 1,1 will be received once the UDP socket is ready for use*
AT+WIPCREATE=1,1,"www.wavecom.com",80 *Note: Create the UDP socket on an arbitrary free local port with peer IP and peer port 80 with communication index = 1 ⇔ WISMO® acts as a UDP client that can send datagram towards the remote entity*	OK *Note: An unsolicited event +WIPREADY: 1,1 will be received once the UDP socket is ready for use*
AT+WIPCREATE=1,1,80,"www.wavecom.com",80 *Note: Create the UDP socket on local port 80 with peer IP and peer port 80 with communication index = 1 ⇔ WISMO® acts as a UDP client and a UDP server : it can send datagram towards the remote entity and receive datagram on the specified local port*	OK *Note: An unsolicited event +WIPREADY: 1,1 will be received once the UDP socket is ready for use*
AT+WIPCREATE=3,1,80,5,8 *Note: Create the TCP server on port 80 with server index = 1 ⇔ WISMO® acts as a TCP server : from now on, it will spawn TCP client sockets from communication index 5 to 8*	OK *Note: An unsolicited event +WIPACCEPT: 1,5 will be received once the TCP server is ready for use*
AT+WIPCREATE=2,1,"IP ADDR",80 *Note: Create the TCP client on port 80 with index=1 ⇔ WISMO® acts as a TCP client : from now on, it can communicate with the remote specified entity through communication index 1*	OK *Note: An unsolicited event +WIPREADY: 2,1 will be received once the TCP client is ready for use*

Notes

The maximum number of sockets can be set to 20 so that the TCP/IP stack can handle 8 UDP sockets, 8 TCP client sockets and 4 TCP servers all at the same time.

Starting a TCP server requires specifying the maximum number of communication sockets that can be spawned. This can be done using the <from idx> and <to idx> parameters. Note that the value set for <to idx> should be equal or more than <from idx>.

The maximum communication socket that can be created using the firmware is 8. Hence, the range for <communication index> and <from idx>, <to idx> is 1-8. Note that the spawned communication socket and the TCP client socket share the same communication index.

It is not possible to create a client socket with AT+WIPCREATE=2, x, y, z when x is already reserved by a server with AT+WIPCREATE=3,<server idx>, <local port>,a,b where a≤x≤b. Similarly, it is not

possible to reserve a range with AT+WIPCREATE=3, <server idx>, <local port>, a, b if one of the TCP client socket indexes between a and b is already reserved, be it by a client or a server range.

The <from idx> and <to idx> are reserved for the server socket till the server socket and the spawned sockets are closed explicitly. So when trying to create a new TCP server socket, the <from idx> and <to idx> should be different from what was used earlier. A parameter used as <from_idx> can't be used as <to_idx> anymore for other TCP server socket creation until spawned sockets with specified <from_idx> and <to_idx> are explicitly closed along with the TCP server socket and vice versa.

When no more communication index is available in the TCP server's range (or no more resources to accept new incoming connections), any peer trying to connect to the server will receive an accept () immediately followed by a shutdown () ("peer close").

Closing a Service +WIPCLOSE

Description

The +WIPCLOSE command is used to close a socket.

Syntax

Action command **AT+WIPCLOSE=<protocol>,<idx>** OK
Read Command **AT+WIPCLOSE?** NONE
Test Command **AT+WIPCLOSE=?** OK
Unsolicited response +WIPPEERCLOSE: <protocol>,<idx>

Parameters and Defined Values

<protocol>:		protocol type
	1	UDP
	2	TCP client
	3	TCP server
<idx>:		socket identifier This parameter is the index of the socket created with the +WIPCREATE

command.

Parameter Storage

None

Possible Errors

The possible error message is displayed only if "AT+CMEE=1" is activated. Otherwise, only "ERROR" is displayed.

"+CMEE" AT error code	Description
802	not enough memory
803	operation not allowed in the current TCP/IP stack state
830	bad index
831	bad state
834	not implemented
837	bad protocol

Examples

Command	Responses
AT+WIPCLOSE=1,1 *Note: Close UDP socket with communication index 1*	OK
AT+WIPCLOSE=2,1 *Note: Close TCP client with communication index 1*	OK
AT+WIPCLOSE=3,1 *Note: Close TCP server with communication index 1*	OK

Notes

After issuing the +WIPCLOSE command, no more data can be sent and received over the socket.

Response “OK” is returned when the +WIPCLOSE command is executed irrespective of whether the socket is active or not.

Service Option Handling +WIPOPT

Description

The +WIPOPT command is used to read and/or to configure different parameters on sockets.

Syntax

- if <action> = 1

Action Command **AT+WIPOPT=<protocol>,<idx>,<action>,<optnum>** OK

- if <action>=2

Action Command **AT+WIPOPT=<protocol>,<idx>,<action>,<optnum>,<optval>** OK
Read Command **AT+WIPOPT?** NONE
Test Command **AT+WIPOPT=?** OK

- if <action> = 1

Unsolicited response +WIPOPT: <protocol>,<optnum>,<optval>

Parameters and Defined Values

<protocol>:		protocol type
	1	UDP
	2	TCP client
	3	TCP server
<idx>:		socket identifier
<action>:		requested operation
	1	read the value of an option
	2	write the value of an option
<optnum>:		option that can be read or written
<optval>:		value of an option

Parameter Storage

None

Possible Errors

The possible error message is displayed only if "AT+CMEE=1" is activated. Otherwise, only "ERROR" is displayed.

"+CMEE" AT error code	Description
800	invalid option
801	invalid option value
803	operation not allowed in the current TCP/IP stack state
830	bad index
834	not implemented
835	option not supported

"+CMEE" AT error code	Description
837	bad protocol
850	unknown reason
860	protocol undefined or internal error
863	protocol delete error
864	protocol list error

Examples

Command	Responses
AT+WIPOPT=2,1,2,8,20 *Note: Set TTL for TCP client.*	OK
AT+WIPOPT=2,1,1,8 *Note: Get TTL for TCP client.*	+WIPOPT: 2,8,20 OK
AT+WIPOPT=3,1,2,9,10 *Note: Set TOS for TCP server*	OK
AT+WIPOPT=3,1,1,9 *Note: Get TOS for TCP server*	+WIPOPT: 3,9,10 OK
AT+WIPOPT=1,1,1,1 *Note: Get peer port for UDP*	+WIPOPT: 1,1,80 OK

Notes

It is possible to change and retrieve the option value using +WIPOPT command only when the socket (given by <idx>) is active, else it returns error.

Refer to the following table for the options that can be applied to UDP, TCP client and TCP server sockets.

opt num	Value format	Option type	Description	UDP	TCP client	TCP server
0	0–65535	WIP_COPT_PORT	Port of the socket	R	R	R
1	0-65535	WIP_COPT_PEER_PORT	Port of the peer socket	R	R	-

opt num	Value format	Option type	Description	UDP	TCP client	TCP server
2	string	WIP_COPT_PEER_STRADDR	Address of the peer socket	R	R	-
6	0-65535	WIP_COPT_NREAD	Number of bytes that can currently be read on the socket default: 0	R	R	-
7	0-1	WIP_COPT_NODELAY	When set to TRUE, TCP packets are sent immediately, even if the buffer is not full enough. When set to FALSE, the packets will be sent either, a) by combining several small packets into a bigger packet b) when the data is ready to send and the stack is idle. default: 0	-	RW	RW
8	1-255	WIP_COPT_TTL	Time-to-leave for packets default: 64	RW	RW	RW
9	0-255	WIP_COPT_TOS	Type of service default: 0	RW	RW	RW

5. Data Exchange for Protocol Services

This section deals with the data exchange for the services over TCP/IP. All the commands required for the data exchange through different services are mentioned in succeeding sections.

Socket Data Exchange +WIPDATA

Description

The +WIPDATA command is used to read/write from/to a socket. On successful execution of the command, the UART switches to data mode. The UART can be switched back to AT mode by sending "+++" with 1 second guard time before and after the sequence. If data is not read using +WIPDATA command, further data will be delayed.

An unsolicited event is received when there is data to read on a socket.

Data can be sent on the sockets using two modes:

- continuous mode
- continuous transparent mode

Continuous Mode

TCP Sockets in Continuous Mode

In continuous mode, an [ETX] character is considered as an end of data. When an [ETX] character is sent on the UART, the TCP socket is shutdown and the peer side is informed of this shutdown with the indication "[CR][LF]SHUTDOWN[CR][LF]" on the UART.

In case an [ETX]/[DLE] character needs to be transmitted as data, it should be preceded by a [DLE] character. Similarly, [ETX]/[DLE] characters received by the TCP/IP stack from the internet are sent to the host through the serial port preceded by a [DLE] character.

To close sockets, switch the UART to AT command mode and use the +WIPCLOSE command.

UDP Sockets in Continuous Mode

UDP is a connectionless protocol and hence there is no way to detect or cause a shutdown. However, an [ETX] character is used to mark the boundaries of datagrams.

All data written on a UDP socket is collected till an [ETX] character is encountered or the maximum size of the datagram[1] is reached and will be sent as a single datagram. Similarly when reading data, all data will be read till an [ETX] character is encountered which indicates the end of the datagram. Note that, in this mode, packet segmentation feature is not supported.

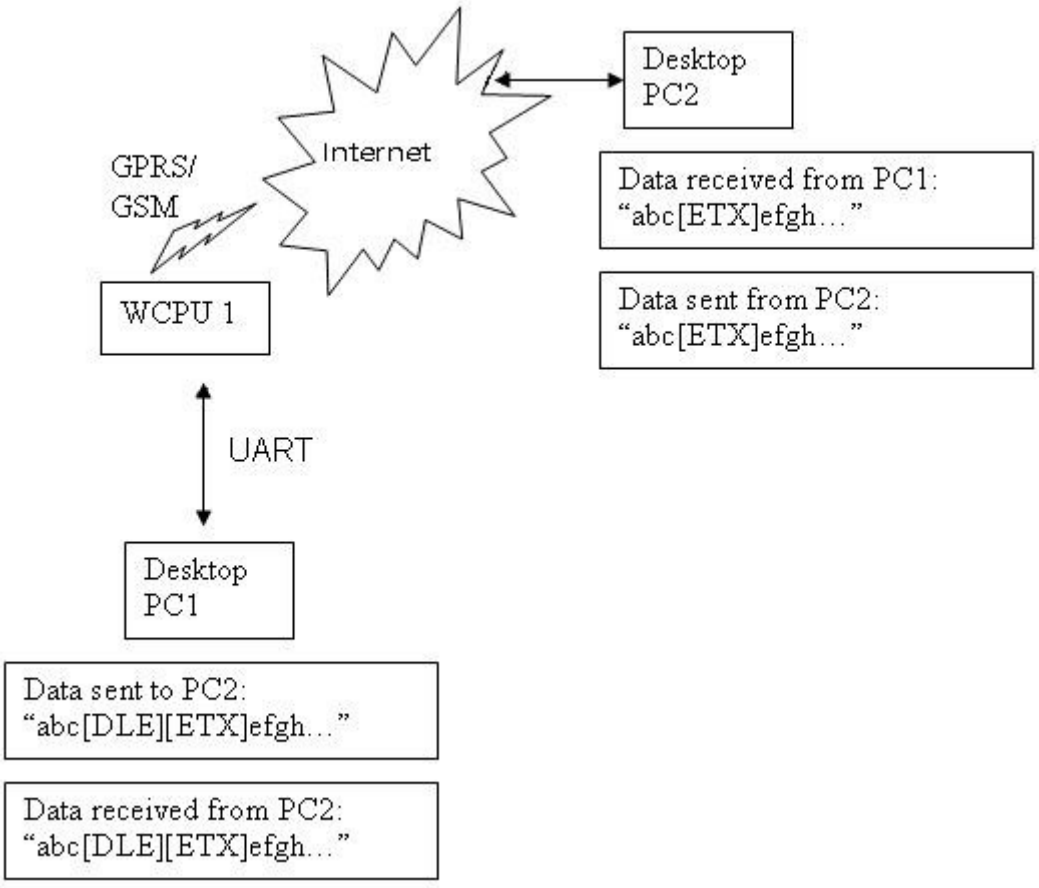
In case an [ETX]/[DLE] character needs to be transmitted, it should be preceded by a [DLE] character similar to the TCP socket.

When the UART leaves DATA mode, either because of a "+++" escape sequence or because of an AT+WIPDATA=1, index, 0 on another UART, the currently unsent data is sent as a single datagram.

[1] *The maximum size of a UDP datagram has been fixed to 5182 Bytes. This limit is an arbitrary one. Note that when the UDP datagram sent is bigger than 5182 Bytes, the whole datagram will be discarded by the WISMO228 module.*

Also note that the smaller the datagram is, the more certain it will reach the aimed destination. UDP is a non-reliable transport layer.

[ETX] Escaping Mechanism



The schematic above explains how [ETX] characters – which have a special meaning in the firmware – are handled on a WISMO®.

On the transmitting side, when [ETX] are not escaped (use case: Desktop PC1 sends data towards the WISMO®. Data contains a non escaped [ETX] (⇔ no [DLE][ETX] sequence), then the [ETX] is not transmitted but an action is done on the WISMO® regarding the concerned socket:

- UDP socket: a non escaped [ETX] marks the boundary of the current datagram to be sent. Datagram is immediately sent and the [ETX] is not sent towards the desktop PC2.
- TCP socket: a non escaped [ETX] causes a TCP shutdown operation on the transmitting direction: peer is informed that the WISMO® will not send any more data on that socket. Usually, peer will shutdown the other way (downlink) and this will result in a "peer close event" on the socket.

On the receiving side, when [ETX] are not escaped (use case: the WISMO® sends data towards Desktop PC1. Data contains a non escaped [ETX] (⇔ no [DLE][ETX] sequence), then the [ETX] means that a special "IP" event occurred on the WISMO® regarding the concerned socket:

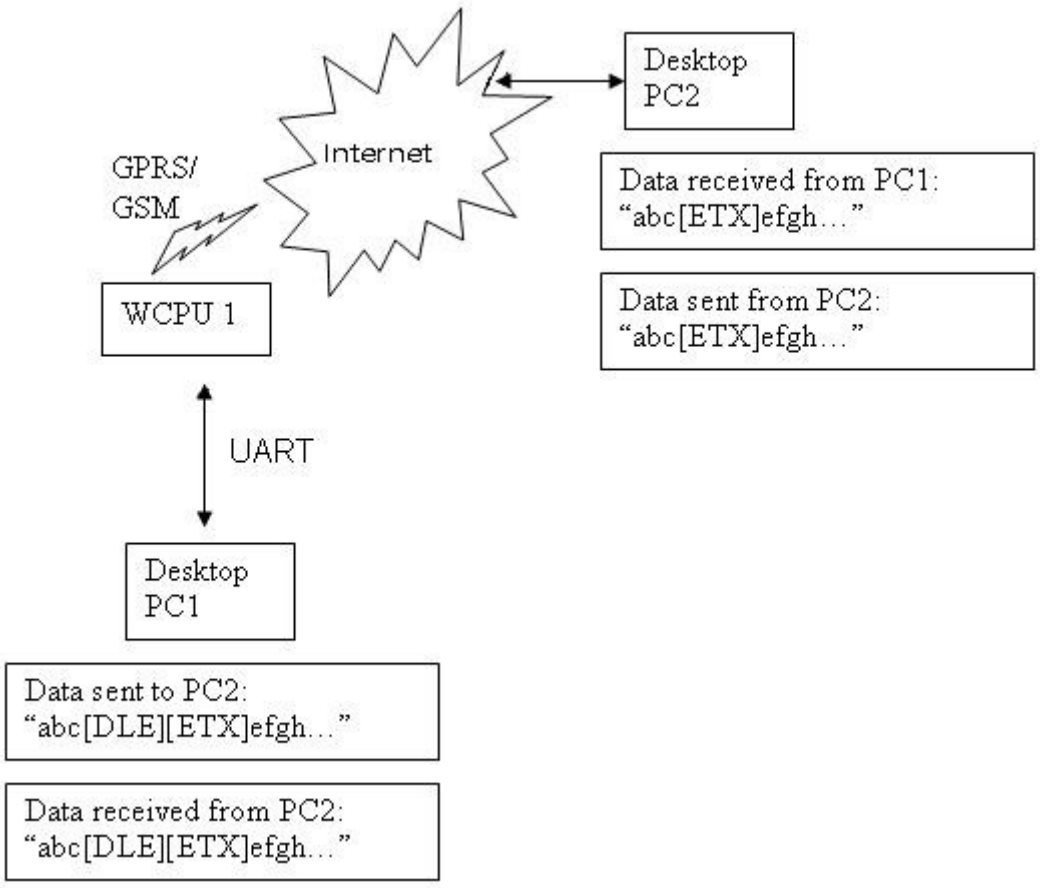
- UDP socket: a non escaped [ETX] signals the boundary of the current received datagram.
- TCP socket: a non escaped [ETX] signals that the peer TCP connected to the TCP unit has shutdown (through the downlink way). Desktop PC1 should then close the uplink socket to totally terminate the TCP "session".

Protocol	UART	IP Network (active socket)
UDP	Data containing a [DLE][ETX] sequence.	Data containing an [ETX].
UDP	[ETX] alone.	Mark the boundary of the UDP Datagram received/to be transmitted.
TCP	Data containing a [DLE][ETX] sequence.	Data containing an [ETX].
TCP	[ETX] alone.	Causes/signals a shutdown operation on TCP socket.

Note: The behaviour is symmetrical in the sense that it applies both on the transmitting and the receiving side of the UART.

[DLE] Escaping Mechanism

A [DLE] character will be sent as data only when it is preceded by another [DLE] character. A single [DLE] character which is not preceded by a [DLE] character will not be transmitted.



The schematic above explains how [DLE] characters – which have a special meaning in the firmware – are handled on the WISMO®.

On the transmitting side, when a [DLE] is not escaped (use case: Desktop PC1 sends data towards the WISMO®. Data contains a non escaped [DLE] (⇔ no [DLE][DLE] sequence), then the [DLE] is not transmitted.

On the transmitting side, when a [DLE] is escaped (use case: Desktop PC1 sends data towards the WISMO®. Data contains an escaped [DLE] (⇔ [DLE][DLE] sequence), then the [DLE] data is transmitted.

On the receiving side (use case: when Desktop PC2 sends data towards the WISMO®. Data contains an escaped [DLE]), the data sent from the WISMO® to Desktop PC1 will contain an escaped [DLE] preceding the [DLE] character (Desktop PC1 receives the [DLE][DLE] character from the WISMO®).

The scenario is the same for both TCP and UDP sockets.

Protocol	UART	IP Network (active socket)
UDP	Data containing a [DLE][DLE] sequence.	Data containing [DLE].
UDP	[DLE] alone.	A single [DLE] is ignored.
TCP	Data containing a [DLE][DLE] sequence.	Data containing [DLE].

Protocol	UART	IP Network (active socket)
TCP	[DLE] alone.	A single [DLE] is ignored.

Continuous Transparent Mode

TCP Sockets in Continuous Transparent Mode

In this mode there is no special meaning associated for [DLE]/[ETX] characters. They are considered as normal data and all the data will be transmitted on the UART.

UDP Sockets in Continuous Transparent Mode

In this mode there is no special meaning associated for [DLE]/[ETX] characters. They are considered as normal data and all data will be transmitted on the UART. In case a [ETX]/[DLE] character is received, it will not be preceded by a [DLE] character before sending it to the UART.

Leaving Continuous/Continuous Transparent Mode

The UART can be switched back to AT mode by sending:

- "+++" with 1 second guard time before and after the sequence
- an AT+WIPDATA=<proto.,<index>,0 on another UART in AT mode

When the UART leaves data mode because of the "+++" escape sequence, the currently unsent data is sent as a single datagram.

Resetting TCP Sockets

A TCP socket is reset when the connection is aborted due to an error on the socket. When the socket is reset, an [ETX] character is sent on the UART to indicate the end of communication. The UART switches to AT mode and "+CME ERROR: 843" is displayed on the UART.

Syntax

Action Command

AT+WIPDATA=<protocol>,<idx>,<mode>[,<send size>,<wait time>]

CONNECT

Read Command **AT+WIPDATA?** NONE
Test Command **AT+WIPDATA=?** OK

- if <protocol> = 1

Unsolicited response +WIPDATA: <protocol>,<idx>,<datagram size>,<peer IP>,<peer port>

Caution: ***Using AT+WIP AT commands, when receiving several UDP datagrams on an IP bearer, +WIPDATA indication is sent once for the first received datagram. Next indication (for next remaining UDP datagram to read) is sent once the first datagram have been read (using +WIPDATA command).***

- if <protocol> = 2

Unsolicited response +WIPDATA: <protocol>,<idx>,<number of readable bytes>

Caution: ***The value returned by <number of readable bytes> indicates that there is some TCP data ready to be read but number of bytes returned might not be reliable. Moreover, using AT+WIP AT commands, when receiving several TCP packets on an IP bearer, +WIPDATA indication is sent once for the first received packet. The next indication (for the next remaining TCP packet to read) is sent after the first packet has been read (using +WIPDATA command).***

Parameters and Defined Values

<protocol>:		socket type
	1	UDP
	2	TCP client
<idx>:		socket identifier
<mode>:		mode of operation
	0	unmap: switch the UART to AT mode.
	1	continuous: switch the UART to data mode.
	2	continuous transparent: switch the UART to data mode. In this mode,

		[DLE]/[ETX] characters are considered as normal data and not special characters.
<send size>:		data packet size: This parameter specifies the size of the data packet that needs to be sent to the peer. This parameter is supported only for UDP continuous transparent mode. range: 8-1460 (default value: 1020)
<wait time>:		timeout for configuring the packet segmentation on the IP network side: This parameter specifies the timeout after which the buffered data will be sent to the peer, irrespective of size of the data packet. This parameter is only supported for UDP continuous transparent mode. range: 1-100 (default value: 2) unit: second

Parameter Storage

None

Possible Errors

The possible error message is displayed only if "AT+CMEE=1" is activated. Otherwise, only "ERROR" is displayed.

"+CMEE" AT error code	Description
831	bad state
837	bad protocol
843	connection reset by peer

Examples

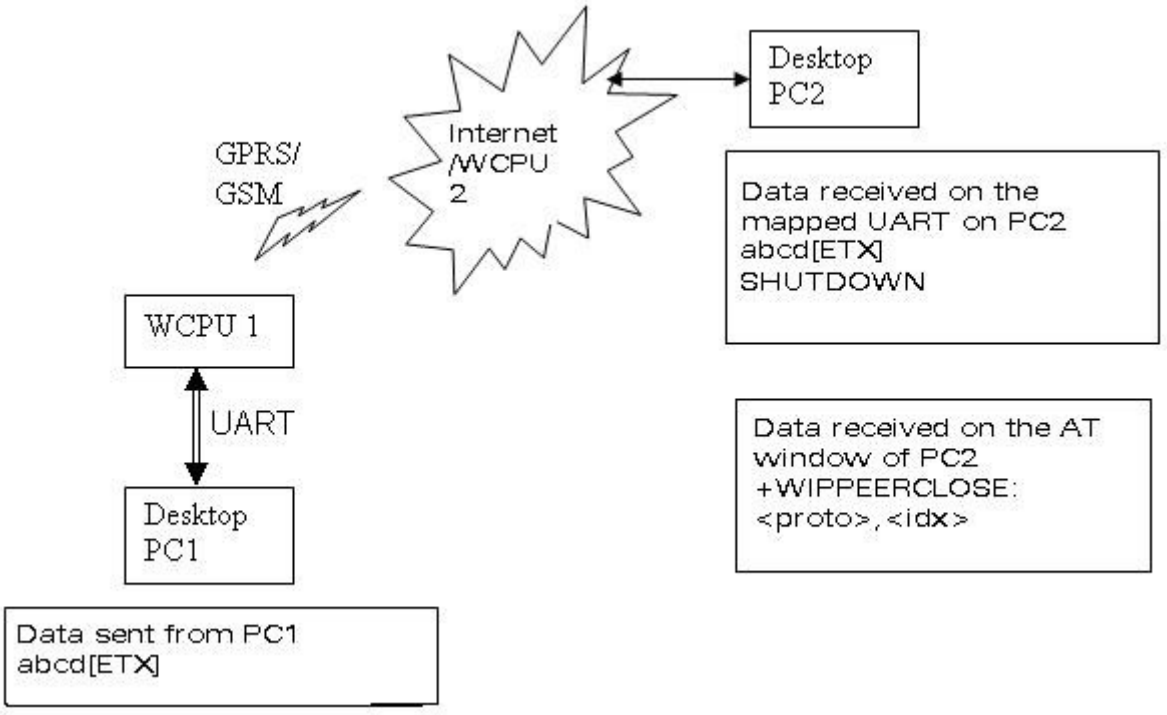
Command	Responses
AT+WIPDATA=2,5,1 *Note: TCP Client with index 5 can send/read data in continuous mode.*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode.*
AT+WIPDATA=2,5,1,10,5 *Note: TCP Client with index 5 can send/read data in continuous mode*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode*
AT+WIPDATA=1,5,1 *Note: UDP with index 5 can send/read data in continuous mode*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode*
AT+WIPDATA=1,5,1 *Note: UDP with index 5 can send/read data in continuous mode*	CONNECT <read/write data> <ETX> OK *Note: [ETX] character indicates end of data*
AT+WIPDATA=1,5,2 *Note: UDP with index 5 can send/read data in continuous transparent mode with default value set for <send size> and <wait time>*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode*
AT+WIPDATA=1,5,2,20,2 *Note: UDP with index 5 can send/read data in continuous transparent mode with <send size> set to 20 and <wait time> set to 2*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode*

Command	Responses
AT+WIPDATA=2,5,1,20,10 *Note: TCP with index 5 can send/read data in continuous mode with <send size> set to 20 and <wait time> set to 10*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode*
AT+WIPDATA=2,5,2,10,5 *Note: TCP with index 5 can send/read data in continuous transparent mode with <send size> set to 10 and <wait time> set to 5*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode*
AT+WIPDATA=2,5,2 *Note: TCP with index 5 can send/read data in continuous transparent mode*	CONNECT <read/write data> +++ OK *Note: +++ sequence causes the UART to switch to AT mode*

Notes

Continuous Mode (Non-Transparent) for a TCP Mapped Socket

If the [ETX] character is sent from the peer, it is considered as an end of data transfer. After sending an [ETX] character, the socket will be shutdown and the peer will be informed of this shutdown by a "[CR][LF]SHUTDOWN[CR][LF]" indication on its UART and the UART will not switch to AT mode. This indicates that no more data can be sent from the host socket, but it can receive data. The schematic below shows the shutdown procedure for a TCP socket:



In the schematic above, a TCP socket is connected. On the transmitting side, data and [ETX] is sent (use case: Desktop PC1 is a WISMO® which sends data to PC2 which is either a PC or also a WISMO®), and the data is received on PC2 and the [ETX] character shuts down the socket on the transmitting side and displays a message "[CR][LF]SHUTDOWN[CR][LF]" on the UART of PC2.

When PC2 is switched back to AT mode, "+WIPPEERCLOSE: <protocol>,<idx>" indication is received indicating that no more data can be sent by PC1 but can read data sent from PC2.

There are different indications received for shutdown and reset for a TCP socket. When a TCP socket is reset, [ETX] character is sent on the UART to indicate the end of communication. The UART switches to AT mode and "+CME ERROR: 843" is displayed on the UART. The reset and shutdown can therefore be distinguished by the indications received on the UART.

Mapping/Unmapping of a Mapped UDP and TCP Socket

When a TCP socket is unmapped and still active, it is possible to map it again in another mode which is different from the previous one without closing the TCP socket.

The UART switches back to AT mode due to the "+++"with 1 second guard time before and after the sequence or by sending an AT+WIPDATA=<proto>,<index>,0 on another UART in AT mode. This applies to both UDP and TCP protocols.

When +++ is issued, the WISMO® switches from DATA mode to AT mode. If ATO command is used to switch the WISMO® back to DATA mode, and:

- +CME ERROR:3 will be received when GPRS bearer is used
- no response is received when GSM bearer is used

To switch the WISMO® back to DATA mode, AT+WIPDATA=x,x,x should be used instead of ATO. After executing the AT+WIPDATA=x,x,x command, "CONNECT" will be received to indicate that the WISMO® is switched back to DATA mode.

Note that un-mapping a socket using the +WIPDATA command with <send size> and <wait time> specified results in "ERROR".

Time Out Mechanisms to Know the State of the Peer TCP Socket

In a TCP server-client connection between two remote devices, if the peer socket is closed down abruptly (e.g. powered off) the peer TCP socket does not get any indication message. This is normal behavior. The TCP protocol uses a timeout mechanism to check the state of the TCP sockets in a TCP socket connection. According to this mechanism, to know the state of the peer TCP socket, the data needs to be sent and wait for the acknowledgement within a specified time period. If the acknowledgement is not received within the specified time out period then the data is retransmitted. But if the time out occurs before receiving the acknowledgement then it implies that the peer TCP socket is closed.

TCP Timeout Period = function (R, N)

Where,

R = Round trip time. This is the time for a TCP packet to go to the remote TCP socket and the time to receive the acknowledgement by the transmitter TCP socket. The typical round trip time is 1 second for GPRS.

N = Number of retransmissions allowed before the time out happens.

Hence, the typical timeout period is 10 minutes depending on the network and also the peer TCP socket localization.

Data needs to be sent to know the state of the peer socket. If acknowledgement is not received within the timeout period then "+CME ERROR: 842" is returned. This indicates that the peer socket is closed.

Please note that the retransmission of the data to the peer TCP socket within the timeout period is managed by the TCP/IP stack.

Packet Segmentation in TCP Socket

The parameters used for packet segmentation can be configured using the +WIPDATA or the +WIPCFG command. If it is not configured using the +WIPDATA command, then the values already

set for option WIP_NET_OPT_TCP_MIN_MSS and AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE will be used.

Note: Any attempt made to set the data packet size to more than twice the value of WIP_NET_OPT_TCP_MIN_MSS using the +WIPDATA command results in "+CME ERROR: 847".

The data sent to a mapped TCP socket through the UART will be buffered before being sent to the peer. This buffered data will be sent to the peer when the:

- total amount of buffered data is twice or more than the preferred segmentation size. The preferred segmentation size is configurable through the "AT+WIPCFG = 2, 4, <size>" (WIP_NET_OPT_TCP_MIN_MSS) or the +WIPDATA command.
- internal timer expires. The timeout period is configurable through the "AT+WIPCFG = 2,12,<time>" (AT_WIP_NET_OPT_PREF_TIMEOUT_VALUE) or +WIPDATA command.
- socket is unmapped, shut down or closed.

In some scenarios, there might be a segmentation of data packets because of timer expiration, network problems, etc. Thus, a single packet of data may be received in more than one packet at the peer.

Packet Segmentation in UDP Sockets

This feature for UDP is supported only in case of continuous transparent mode. If the +WIPDATA command is executed in continuous mode to use this feature, "ERROR" will be returned. The parameters used for packet segmentation can be configured using the +WIPDATA command. In case they are not configured using the +WIPDATA command, the default value of these parameters will be used.

In some scenarios, there might be a segmentation of data packets because of timer expiration, network problems, etc. Thus, a single packet of data may be received in more than one packet at the peer.

6. Ping Services

PING command +WIPPING

Description

The +WIPPING command is used to configure different PING parameters and to send PING requests. An unsolicited response is displayed each time a "PING" echo event is received or a timeout expires.

Syntax

Action Command
`AT+WIPPING=<host>,[<repeat>],[<interval>],[<timeout>],[<nwrite>],[<ttl>]`
`OK`

Read Command **`AT+WIPPING?`** `OK`
Test Command **`AT+WIPPING=?`** `OK`
Unsolicited response `+WIPPING:<timeout_expired>,<packet_idx>,<response_time>`

Parameters and Defined Values

<host>:	host name or IP address string
<repeat>:	number of packets to send range: 1-65535 (default value:1)

<interval>:		number of milliseconds between packets range: 1-65535 (default value:2000)
<timeout>:		number of milliseconds before a packet is considered lost range: 1-65535 (default value:2000)
<ttl>:		IP packet Time To Live Default value is set by WIP_NET_OPT_IP_TTL +WIPCFG option range : 0-255
<nwrite>:		size of user data packets (excluding the 28 bytes IP header + ICMP header) range : 1-1472 (default value:64)
<timeout_expired>:		PING result
	0	PING response received before <timeout>
	1	<timeout> expired before the response was received
<packet_idx>:		packet index in the sequence
<response_time>:		PING response time in millisecond

Parameter Storage

None

Possible Errors

The possible error message is displayed only if "AT+CMEE=1" is activated. Otherwise, only "ERROR" is displayed.

"+CMEE" AT error code	Description
800	invalid option
801	invalid option value
819	error on Ping channel

Examples

Command	Responses
AT+WIPPING="www.wavecom.com" *Note: Ping "www.wavecom.com"*	OK +WIPPING: 1,0,0 *Note: Ping "www.wavecom.com failed : timeout expired.*
AT+WIPPING="192.168.0.1" *Note: Ping "192.168.0.1"*	OK +WIPPING: 0,0,224 *Note: Ping "192.168.0.1 succeeded. Ping response received in 224ms.*
AT+WIPPING="192.168.0.1",2,2000,1000 *Note: Send 2 successive ping requests to "192.168.0.1". Each Ping is 2000 ms, timeout is set to 1000 ms (if ping response time is more than 1000 ms, then timeout expires).*	OK +WIPPING: 0,0,880 +WIPPING: 1,1,xxxx *Note: Ping "192.168.0.1 succeeded. First Ping response received in 880 ms. Second one was not received before specified timeout (1000 ms) ⇨ timeout expired.*

7. Examples of Application

TCP Socket

TCP Server Socket

Using GPRS Bearer

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name (<login>)*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password (<password>)*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=3,1,80,5,8	*//create the server on port 80, idx = 1. The server is // listening for connection request on port 80. //Spawned sockets will be given the index 5, 6, 7 and 8. //It will accept connection request until it has no more //sockets left.*
OK	
+WIPACCEPT: 1,5	*//unsolicited: the server accepted a connection resulting //TCP client on idx 5.*
AT+WIPDATA=2,5,1	*//exchange data on socket index 5*
CONNECT	
...	*//read, write*
+++	*//switch to AT mode*
OK	
AT+WIPCLOSE=2,5	*//close the TCP client socket index 5*
OK	

Using GSM Bearer

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,5	*//open GSM bearer*
OK	
AT+WIPBR=2,5,2,"Phone number"	*//set phone number for GSM bearer*
OK	
AT+WIPBR=2,5,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,5,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,5,0	*//start GSM bearer*
OK	
AT+WIPCREATE=3,1,80,5,8	*//create the server on port 80, idx = 1. The server is //listening for connection request on port 80.Spawned //sockets will be given the index 5, 6, 7 and 8. It will //accept connection request until it has no more sockets //left.*
OK	
+WIPACCEPT: 1,5	*//unsolicited: the server accepted a connection resulting //TCP client on idx 5*
AT+WIPDATA=2,5,1	*//exchange data on socket idx 5*
CONNECT	
...	*//read, write*
+++	*//switch to AT mode*
OK	
AT+WIPCLOSE=2,5	*//close the TCP client socket index 5*
OK	

TCP Client Socket

Using GPRS Bearer

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=2,1,"ip addr",80	*//create a TCP client towards peer IP device @ "ip addr", //port 80.*
OK	*//all parameters and iP stack behavior are OK.*
+WIPREADY: 2,1	*//unsolicited: the TCP client socket is connected to the //peer*
AT+WIPDATA=2,1,1	*//exchange data on socket idx 1:*
CONNECT	
...	*//read, write*
+++	*//switch to AT mode*
OK	
AT+WIPCLOSE=2,1	*//close the TCP client socket index 1*
OK	

Using GSM Bearer

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,5	*//open GSM bearer*
OK	
AT+WIPBR=2,5,2,"Phone number"	*//set phone number for GSM bearer*
OK	
AT+WIPBR=2,5,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,5,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,5,0	*//start GSM bearer*
OK	
AT+WIPCREATE=2,1,"ip addr",80	*//create a TCP client towards peer IP device @ "ip addr", //port 80*
OK	*//all parameters and IP stack behavior are OK*
+WIPREADY: 2,1	*//unsolicited: the TCP client socket is connected to the //peer*
AT+WIPDATA=2,1,1	*//exchange data on socket idx 1*
CONNECT	
…	*//read, write*
+++	*//switch to AT mode*
OK	
AT+WIPCLOSE=2,1	*//close the TCP client socket index 1*
OK	

UDP Socket

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=1,1,80,"www.wavecom.com",80	*//create a UDP client towards peer IP device @ //"www.wavecom.com" , port 80*
OK	*//all parameters and IP stack behavior are OK*
+WIPREADY: 1,1	*//unsolicited: the UDP client socket is "pseudo" connected //to the peer (no //real connection is //UDP)*
AT+WIPDATA=1,1,1	*//exchange data on socket idx 1:*
CONNECT	
…	*//read, write*
+++	*//switch to AT mode*
OK	
AT+WIPCLOSE=1,1	*//close the UDP socket index 1*
OK	
AT+WIPCREATE=1,1,1234	*//start a UDP server and listen for datagram on port 1234*
OK	*//all parameters and IP stack //behavior are OK*
+WIPREADY: 1,1	*//unsolicited: the UDP client socket is "pseudo" connected //to the peer (no real connection is UDP)*

+WIPDATA: 1,1,25,"192.168.0.2",2397	*//one datagram is ready to be read : it was sent from //192.168.0.2 on port 2397 and is composed of 25 bytes*
AT+WIPDATA=1,1,1	
CONNECT	
abcedghijklmnopqrstuvwxyz[ETX]	*//here 25 bytes + the [ETX] character (marking the bound //of the datagram) have been read.*
OK	*//here UART is back to AT command mode. If some other //remote IP devices sent one or more datagrams while //reading for the first one, then a new datagram indication //is received*
+WIPDATA: 1,1,50,"192.168.0.4",58	*//one datagram is ready to be read : it was sent from //192.168.0.4 on port 58 and is composed of 50 bytes*
AT+WIPDATA=1,1,1	
CONNECT	
abcedghijklmnopqrstuvwxyzabcedghijklm nopqrstuvwxyz [ETX]	*//here 25 bytes + the [ETX] character (marking the bound //of the datagram) have been read.*

PING

```
AT+WIPCFG=1                                  //start IP stack
OK

AT+WIPBR=1,6                                 //open GPRS bearer
OK

AT+WIPBR=2,6,11,"APN name"                   //set APN name of GPRS bearer
OK

AT+WIPBR=2,6,0,"user name"                   //set user name
OK

AT+WIPBR=2,6,1,"passwd"                      //set password
OK

AT+WIPBR=4,6,0                               //start GPRS bearer
OK

AT+WIPPING="192.168.0.1"                     //start PING session
OK
+WIPPING:0,0,224
```

Creating a TCP Server, Spawning the Maximum TCP Sockets (for the Configured Server)

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=3,1,80,5,6	*//create the server on port 80, idx = 1. The server is //listening for connection request on port 80.Spawned //sockets will be given the index 5 or 6. It will accept //connection request until it has no more socket left.*
OK	
+WIPACCEPT: 1,5	*//unsolicited: the server accepted a connection //resulting TCP client on idx 5.*
+WIPACCEPT: 1,6	*//unsolicited: the server accepted a connection //resulting TCP client on idx 6.*
AT+WIPCLOSE=2,5	*//close the spawned TCP client socket index 5.*
OK	*//now if the peer device try to connect to the server it //shall receive an accept () immediately followed by an shutdown() (connection reset by peer)*

Creating a Server and Trying to Create a TCP Client/Server on a Reserved Index (reserved by the Server) will Fail

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=3,2,80,1,2	*//create the server on port 80, idx=2. The server is //listening for connection request on port 80. Spawned //sockets will be given the index 1 or 2.It will accept //connection request until has nor more socket left.*
OK	
AT+WIPCREATE=2,3,"198.168.0.1",80	*//create a TCP client towards peer IP device @ //"198.168.0.1", port 80,*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 2,3	*//unsolicited: the TCP client socket is connected to the //peer.*
+WIPACCEPT: 2,1	*//unsolicited: the server index accepted a connection; //resulting TCP client on idx 1*
AT+WIPDATA=2,3,1	*//exchange data on socket index 3*
CONNECT	
AT+WIPDATA=2,1,1	*//exchange data on socket index 1*
CONNECT	
[ETX]	*//send unescaped ETX character*

+WIPPEERCLOSE: 2,3	*//unsolicited: peer socket is closed*
AT+WIPCLOSE=3,1 OK	*//close TCP server socket index 1*
AT+WIPCREATE=3,2,81,2,3	*//create the server on port 81, idx=2 and from_idx=2 //and to_idx=3*
+CME ERROR:845	*//TCP client socket with idx 2 was reserved by the //previous server socket and it was not closed //explicitly. Hence error is returned.*

Failed Creation of a TCP Client and Failed Creation of a TCP Server with Index Range Containing a TCP Client

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=2,1,"198.168.0.1",80	*//create a TCP client towards peer IP device @ //"198.168.0.1", port 80*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 2,1	*//unsolicited: the TCP client socket is connected to the //peer.*
AT+WIPCREATE=3,2,80,1,2	*//create the server on port 80, idx=2. Range //requested contains the already used index "1" and //hence error is returned.*
+CME ERROR: 845	

Creating 8 UDP Sockets, 8 TCP Clients and 4 TCP Servers

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=1,1,55,"192.168.0.1",75	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 75.*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 1,1	*//unsolicited: the UDP client socket is "pseudo //"connected to the peer (no real connection in UDP)*
AT+WIPCREATE=1,2,56,"192.168.0.1",76	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 76.*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 1,2	*//unsolicited: the UDP client socket is "pseudo //"connected to the peer (no real connection is UDP)*
AT+WIPCREATE=1,3,57,"192.168.0.1",77	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 77.*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 1,3	*//unsolicited: the UDP client socket is "pseudo //"connected to the peer (no real connection is UDP)*
AT+WIPCREATE=1,4,58,"192.168.0.1",78	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 78.*
OK	*//all parameters and IP stack behavior are OK*

```
+WIPREADY: 1,4                          //unsolicited: the UDP client socket is "pseudo
                                        //"connected to the peer (no real connection in UDP)

AT+WIPCREATE=1,5,59,"192.168.0.1",79    //create a UDP client towards peer IP device @
                                        //"192.168.0.1", port 79.

OK                                      //all parameters and IP stack behavior are OK

+WIPREADY: 1,5                          //unsolicited: the UDP client socket is "pseudo
                                        //"connected to the peer (no real connection in UDP)

AT+WIPCREATE=1,6,60,"192.168.0.1",80    //create a UDP client towards peer IP device @
                                        //"192.168.0.1", port 80.

OK                                      //all parameters and IP stack behavior are OK

+WIPREADY: 1,6                          //unsolicited: the UDP client socket is "pseudo
                                        //"connected to the peer (no real connection in UDP)

AT+WIPCREATE=1,7,61,"192.168.0.1",81    //create a UDP client towards peer IP device @
                                        //"192.168.0.1", port 81

OK                                      //all parameters and IP stack behavior are OK

+WIPREADY: 1,7                          //unsolicited: the UDP client socket is "pseudo
                                        //"connected to the peer (no real connection in UDP)

AT+WIPCREATE=1,8,62,"192.168.0.1",82    //create a UDP client towards peer IP device @
                                        //"192.168.0.1", port 82.

OK                                      //all parameters and IP stack behavior are OK

+WIPREADY: 1,8                          //unsolicited: the UDP client socket is "pseudo
                                        //"connected to the peer (no real connection in UDP)

AT+WIPCREATE=1,9,63,"192.168.0.1",83
+CME ERROR: 830
                                        //8 UDP sockets have been created and hence 9th
                                        //attempt fails

AT+WIPCREATE=3,1,80,1,1                 //create one server on port 80, idx = 1. One TCP
                                        //client socket is reserved on index 1
OK

AT+WIPCREATE=3,2,81,2,2                 //create one server on port 81, idx = 2. One TCP
                                        //client socket is reserved on index 2
OK

AT+WIPCREATE=3,3,82,3,3                 //create one server on port 82, idx = 3. One TCP
                                        //client socket is reserved on index 3
OK

AT+WIPCREATE=3,4,83,4,4                 //create one server on port 83, idx = 4. One TCP
                                        //client socket is reserved on index 4
OK

AT+WIPCREATE=3,5,84,5,5                 //4 TCP servers have been created and hence
                                        //creation of 5th TCP server socket fails
+CME ERROR: 830
```

AT+WIPCREATE=2,1,"192.168.0.1",80	*//create a TCP client socket towards peer IP device //@ "192.168.0.1", port 80. Index 1 is reserved by //server index and hence error is returned.*
+CME ERROR: 845	
	//4 reserved TCP client sockets have been spawned //by their TCP server.
+WIPACCEPT: 1,1	*//unsolicited: the server index 1 accepted a //connection; resulting TCP client on idx 1*
+WIPACCEPT: 2,2	*//unsolicited: the server index 2 accepted a //connection; resulting TCP client on idx 2*
+WIPACCEPT: 3,3	*//unsolicited: the server index 3 accepted a //connection; resulting TCP client on idx 3*
+WIPACCEPT: 4,4	*//unsolicited: the server index 4 accepted a //connection; resulting TCP client on idx 4*
AT+WIPCREATE=2,5,"192.168.0.1",80	*//create a TCP client towards peer IP device @ //"192.168.0.1", port 80.*
OK	*//all parameters and IP stack behavior are OK*
+WIPREADY: 2,5	*//unsolicited: the TCP client socket is connected to the //peer.*
AT+WIPCREATE=2,6,"192.168.0.1",80	*//create a TCP client towards peer IP device @ //"192.168.0.1", port 80.*
OK	*//all parameters and IP stack behavior are OK*
+WIPREADY: 2,6	*//unsolicited: the TCP client socket is connected to the //peer*
AT+WIPCREATE=2,7,"192.168.0.1",80	*//create a TCP client towards peer IP device @ //"192.168.0.1", port 80*
OK	*//all parameters and IP stack behavior are OK*
+WIPREADY: 2,7	*//unsolicited: the TCP client socket is connected to the //peer*
AT+WIPCREATE=2,8,"192.168.0.1",80	*//create a TCP client towards peer IP device @ //"192.168.0.1", port 80.*
OK	*//all parameters and IP stack behavior are OK*
+WIPREADY: 2,8	*//unsolicited: the TCP client socket is connected to the //peer*
AT+WIPCREATE=2,8,"192.168.0.1",80	*//create a TCP client towards peer IP device @ //"192.168.0.1", port 80. Index 8 is already used and //corresponds to an active socket.*
+CME ERROR: 840	
AT+WIPCREATE=2,9,"192.168.0.1",80	*//create a TCP client towards a peer IP device @ //"192.168.0.1", port 80. Index 9 is forbidden.*
+CME ERROR: 830	

Trying to Create 8 UDP Sockets, 8 TCP Client Sockets and 4 TCP Server Sockets

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=1,1,55,"192.168.0.1",75	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 75.*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 1,1	*//unsolicited: the UDP client socket is "pseudo //"connected to the peer (no real connection in // UDP)*
AT+WIPCREATE=1,2,56,"192.168.0.1",76	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 76.*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 1,2	*//unsolicited: the UDP client socket is "pseudo //"connected to the peer (no real connection in // UDP)*
AT+WIPCREATE=1,3,57,"192.168.0.1",77	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 77.*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 1,3	*//unsolicited: the UDP client socket is "pseudo //"connected to the peer (no real connection in // UDP)*
AT+WIPCREATE=1,4,58,"192.168.0.1",78	*//create a UDP client towards peer IP device @ //"192.168.0.1", port 78.*
+CME ERROR: 838	*//maximum 3 sockets can be created as the //MAX_SOCK_NUM value has been changed to 3. //Hence an attempt to create a fourth socket returns //error.*

Creating TCP Client and Server Sockets in the Same WISMO®

AT+WIPCFG=1	*//start IP stack*
OK	
AT+WIPBR=1,6	*//open GPRS bearer*
OK	
AT+WIPBR=2,6,11,"APN name"	*//set APN name of GPRS bearer*
OK	
AT+WIPBR=2,6,0,"user name"	*//set user name*
OK	
AT+WIPBR=2,6,1,"passwd"	*//set password*
OK	
AT+WIPBR=4,6,0	*//start GPRS bearer*
OK	
AT+WIPCREATE=3,2,80,1,2	*//create the server on port 80, idx=2. The server is //listening for connection requests on port 80. //Spawned sockets will be given the index 1 or 2. It //will accept connection request until it has nor more //sockets left.*
OK	
AT+WIPCREATE=2,3,"198.168.0.1",80	*//create a TCP client towards peer IP device @ //"198.168.0.1", port 80,*
OK	*//all parameters and IP stack behavior are OK.*
+WIPREADY: 2,3	*//unsolicited: the TCP client socket is connected to the //peer.*
+WIPACCEPT: 2,1	*//unsolicited: the server index accepted a connection; //resulting TCP client on idx 1*
AT+WIPDATA=2,3,1	*//exchange data on socket index 3*
CONNECT	
abc+++	*//data sent to socket index 1 and switched to AT //mode by giving +++*
OK	
AT+WIPDATA=2,1,1	*//exchange data on socket index 1*
CONNECT	

8. Error Codes

"+CMEE" AT Error Code	Description
3	operation not allowed
800	invalid option
801	invalid option value
802	not enough memory
803	operation not allowed in the current TCP/IP stack state
804	device already open
805	network interface not available
806	operation not allowed on the considered bearer
807	bearer connection failure : line busy
808	bearer connection failure : no answer
809	bearer connection failure : no carrier
810	bearer connection failure : no sim card present
811	bearer connection failure : sim not ready (no pin code entered, …)
812	bearer connection failure : GPRS network failure
813	bearer connection failure : PPP LCP negotiation failed
814	bearer connection failure : PPP authentication failed
815	bearer connection failure : PPP IPCP negotiation failed
816	bearer connection failure : PPP peer terminates session
817	bearer connection failure : PPP peer does not answer to echo request
818	incoming call refused
819	error on Ping channel
820	error writing configuration in non-volatile memory
821	error reading configuration in non-volatile memory
822-829	reserved for future use
830	bad index
831	bad state
832	bad port number
833	bad port state
834	not implemented
835	option not supported
836	memory allocation error

"+CMEE" AT Error Code	Description
837	bad protocol
838	no more free socket
839	error during channel creation
840	UDP/TCP socket session is already active
841	peer closed
842	destination host unreachable (whether host unreachable, Network unreachable or response timeout)
843	connection reset by peer
844	stack already started
845	attempt is made to reserve/create a client socket which is already reserved/opened by TCP server/client
846	internal error
847	bearer connection failure: WIP_BOPT_GPRS_TIMEOUT time limit expired before GPRS bearer connected
848	impossible to connect to the bearer
849	connection to the bearer has succeeded but a problem has occurred during the data flow establishment
850	unknown reason
851	Incorrect number of parameters submitted
852-859	reserved for future use
860	protocol undefined or internal error
861	user name rejected by server
862	password rejected by server
863	protocol delete error
864	protocol list error
865	authentication error
866	server not ready error
869-879	reserved for future use

9. List of Abbreviations

Abbreviation	Definition
APN	Access Point Name
ASCII	American Standard Code for Information Interchange
AT	ATtention
BCC	Blind Carbon Copy
CC	Carbon Copy
CHAP	Challenge Handshake Authentication Protocol
CHV	Card Holder Verification
CID	Context IDentifier
CMUX	Converter Multiplexer
CPU	Central Processing Unit
DNS	Domain Name System
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile communicatio006E
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
M	Mandatory
MS	Mobile Station
MSCHAP	MicroSoft Challenge Handshake Authentication
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
N/A	Not Applicable
NU	Not Used

O	Optional
OS	Operating System
PAP	Password Authentication Protocol
PDP	Packet Data Protocol
PIN	Personal Identity Number
POP3	Post Office Protocol
PPP	Point-to-Point Protocol
SIM	Subscriber Information Module
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
TOS	Type Of Service
TTL	Time To Live
UART	Universal Asynchronous Receiver Transmitter
UDP	User Data Protocol
URL	Uniform Resource Locator
WIP	Wavecom Internet Protocol

SIERRA WIRELESS
HEART OF THE WIRELESS MACHINE®